Module 8: Portfolio Milestone

Online Shopping Cart

Nolan Byrnes

CSC500 – Principals of Programming

Colorado State University – Global Campus

Professor Steven Evans

March 4, 2022

**Source Code:**

from datetime import date, datetime

def main():

name = input("What is your name?\n")

current\_date = ask\_for\_date("What is today's date?\n")

print("Customer Name:",name)

print("Today's Date:",current\_date.strftime("%B %d, %Y"))

cart = ShoppingCart(name, current\_date)

shopping\_cart\_menu(cart)

def get\_item\_details(item):

item.item\_name = input("Enter the item name:\n")

item.description = input("Enter the Description of the Item:\n")

item.item\_price = ask\_for\_number("float", "Enter the item price:\n")

item.item\_quantity = ask\_for\_number("int", "Enter the item quantity:\n")

return item

def shopping\_cart\_menu(ShoppingCart):

while True:

print("\na - Add item to cart")

print("r - Remove Item from cart")

print("c - Change item quantity")

print("i - Output items' descriptions")

print("o - Output shopping cart")

print("q - Quit\n")

menu\_selection = input("Choose an Option:\n")

if menu\_selection == "q":

break

elif menu\_selection == "a":

ShoppingCart.add\_item(get\_item\_details(ItemToPurchase()))

elif menu\_selection == "r":

item\_to\_remove = input("Which Item would you like to remove?\n")

ShoppingCart.remove\_item(item\_to\_remove)

elif menu\_selection == "c":

item\_to\_edit = ItemToPurchase()

item\_name = input("Which Item do you want to change the quantity for?\n")

new\_item\_quantity = ask\_for\_number("int", "What is the new Item Quantity?\n")

item\_to\_edit.item\_name = item\_name

item\_to\_edit.item\_quantity = new\_item\_quantity

ShoppingCart.modify\_item(item\_to\_edit)

elif menu\_selection == "i":

ShoppingCart.print\_descriptions()

elif menu\_selection == "o":

ShoppingCart.print\_total()

else:

print("Please enter a valid menu selection.\n")

print("GoodBye")

def ask\_for\_number(number\_type, text):

not\_number = True

while not\_number:

res = input(text)

try:

if number\_type == "int":

return int(res)

elif number\_type == "float":

return float(res)

except ValueError:

not\_number = True

if number\_type == "int":

print("Please provide an integer as your response.\n")

elif number\_type == "float":

print("Please provide a float as your response.\n")

def ask\_for\_date(text):

not\_date = True

while not\_date:

date\_input = input(text)

try:

if "/" in date\_input:

current\_date = datetime.strptime(date\_input, "%m/%d/%Y")

return current\_date

elif "-" in date\_input:

current\_date = datetime.strptime(date\_input, "%m-%d-%Y")

return current\_date

else:

print("Please insert a date using m/d/yyyy or m-d-yyyy format.")

except ValueError:

print("Please insert a date using m/d/yyyy or m-d-yyyy format.")

class ShoppingCart:

def \_\_init\_\_(self, name, cartDate):

self.customer\_name = name

self.current\_date = cartDate

self.cart\_items = []

pass

def add\_item(self, ItemToPurchase):

self.cart\_items.append(ItemToPurchase)

pass

def remove\_item(self, item\_name):

try:

item = next(cart\_item for cart\_item in self.cart\_items if cart\_item.item\_name == item\_name)

self.cart\_items.remove(item)

except StopIteration:

print("Item not found in cart. Nothing removed.")

pass

def modify\_item(self, ItemToPurchase):

try:

item = next(cart\_item for cart\_item in self.cart\_items if cart\_item.item\_name == ItemToPurchase.item\_name)

if ItemToPurchase.description != "":

item.description = ItemToPurchase.description

if ItemToPurchase.item\_price != 0.00:

item.item\_price = ItemToPurchase.item\_price

if ItemToPurchase.item\_quantity != 0:

item.item\_quantity = ItemToPurchase.item\_quantity

except StopIteration:

print("Item not found in cart. Nothing modified.")

pass

def get\_num\_items\_in\_cart(self):

num\_items = 0

for item in self.cart\_items:

num\_items += item.item\_quantity

return num\_items

def get\_cost\_of\_cart(self):

total = 0

for i in self.cart\_items:

total += i.item\_price \* i.item\_quantity

return total

def print\_total(self):

if len(self.cart\_items)==0:

print("SHOPPING CART IS EMPTY")

else:

print(self.customer\_name + "'s Shopping Cart - " + self.current\_date.strftime("%B %d, %Y"))

print("Number of Items:", self.get\_num\_items\_in\_cart())

for item in self.cart\_items:

item.print\_item\_cost()

print("Total: $" + str(self.get\_cost\_of\_cart()))

pass

def print\_descriptions(self):

print(self.customer\_name + "'s Shopping Cart - " + str(self.current\_date) )

print("Item Descriptions")

for item in self.cart\_items:

print(item.item\_name + ": " + item.description)

pass

class ItemToPurchase:

def \_\_init\_\_(self):

self.item\_name = "none"

self.item\_price = 0.00

self.item\_quantity = 0

self.description = ""

pass

def print\_item\_cost(self):

subtotal = self.item\_quantity \* self.item\_price

print(self.item\_name, self.item\_quantity, "@", "$" + str(self.item\_price), "=", "$" + str(subtotal))

return subtotal

if \_\_name\_\_ == '\_\_main\_\_' : main()

**Asking for User’s Name and Date:**

Text

Description automatically generated

**Add Items to Cart:**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

**Output Shopping Cart:**

Text

Description automatically generated

**Change Quantity:**

Text

Description automatically generated

**Remove Item:**

Text

Description automatically generated

**Output Shopping Cart:**

Graphical user interface, text

Description automatically generated

**Output Item Descriptions:**

Text

Description automatically generated

**Removing Item not in Cart:**

Text

Description automatically generated

**Change Item Quantity Not in Cart:**

Text

Description automatically generated

**Invalid Menu Selection**

Text

Description automatically generated

**Removing Rest of Items:**

Text

Description automatically generated

Text

Description automatically generated

**Displaying Empty Cart:**

Text

Description automatically generated

**Quit:**

Graphical user interface, text

Description automatically generated